بسمه تعالی

تمرین سری دوازدهم درس یادگیری عمیق

پوریا محمدی نسب

۴۰۰۷۲۲۱۳۸

فهرست

سوال ۱.

مقاله ۱ را مطالعه کنید و خلاصه‌ای از آن بنویسید و در آن به سوالات زیر پاسخ دهید.

الف) چالش های اصلی مسئله را بیان کنید و به تفسیر آن ها بپردازید.

ب) راه حل روش‌های پیشین برای حل این چالش‌ها چه بوده است؟

ج) چه راهکارهایی برای ساخت فضای جستجو وجود دارد؟ به توضیح هر یک بپردازید.

د) راه حل این مقاله برای حل چالش‌های گفته شده چه بوده است ؟

ه) تابع Score گفته شده در مقاله را شرح دهید. مهم ترین مشخصه این تابع چیست؟

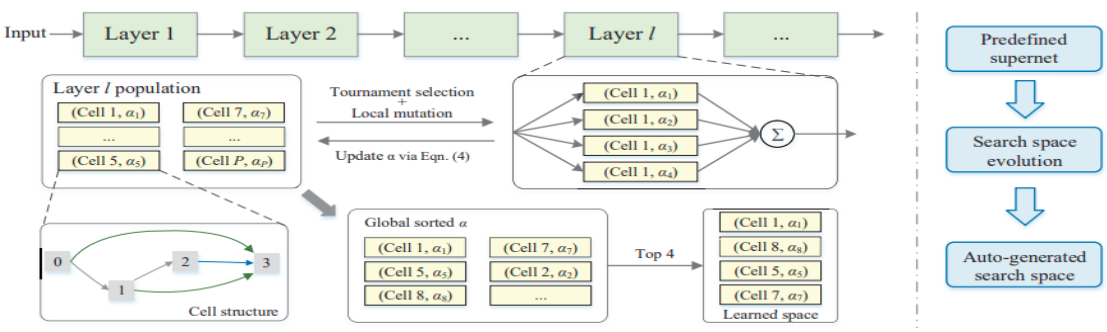
و) پس از مطالعه این مقاله چه راهکار عملیاتی برای بهبود عملکرد آن میتوانید ارائه دهید؟

در این مقاله سعی شده است نسبت به سایر الگوریتم های جستجوی عصبی معماری دخالت و تجربیات انسان را کمتر استفاده کند و یک الگوریتم خودکار با استقلال بیشتری نسبت به سایر الگوریتم ها معرفی کند. دو چالش اساسی با کمرنگ شدن دخالت انسان در فرایند جستجو بیش میاید که عبارتند از انفجار پیچیدگی فضای جستجو و هزینه محاسباتی بسیار بالا برای ارزیابی کیفیت فضاهای جستجوی مختلف. برای حل این دو مشکل در این مقاله یک فریم ورک مشتق بذیر تکاملی به اسم AutoSpace معرفی می‌شود که فضای جستجو را با تکنیک تابع امتیاز دهی مشتق پذیز و یک معماری مرجع به یک فضای بهینه تبدیل میکند.

به طور کلی الگوریتم های NAS در سه مرحله خلاصه می‌شوند: ۱. طراحی یک فضای جستجو با مشخص کردن عملگرهای ابتدایی آن. ۲. استفاده از یک الگوریتم جستجو برای کاوش در سطح فضای جستجو و انتخاب عملگرها در آن برای ساخت یک مدل کاندید ۳. پیاده سازی یک استراتژی ارزیابی برای تأیید عمل‌کرد مدل انتخابی. البته الگوریتم هایی اخیراً معرفی شدند که در دو مرحله ی جستجوی مدل و ارزیابی مدل خلاصه میشوند.

بیشتر الگوریتم های NAS قبلی به دنبال یافتن یک الگوریتم جستجوی مناسب بودند در حالی که طراحی فضای جستجو در این مقالات بسیار کمتر مورد بحث قرار گرفته است و معمولاً از طراحی دستی فضای جستجو استفاده میکنند. یکی از چالش ها و انگیزه‌های این مقاله نیز همین طراحی فضای جستجو است.

مدل پیشنهادی این مقاله در سه مرحله زیر فضای مسئله را یاد میگیرد. شکل زیر نشان دهنده این مراحل میباشد.



روش پیشنهادی تولید خودکار فضای جستجو

یک الگوریتم تکاملی به طور معمول ۳ مرحله دارد: ۱. تولید جمعیت ۲. ارزیابی هر فرد در جمعیت با استفاده از تابع امتیاز دهی. ۳. تولید جمعیت جدید از جمعیت قبلی و جهش در بعضی افراد جدید جامعه.

۲ مرحله نهایی ذکر شده در الگوریتم های تکاملی به صورت تکرارشونده هستند تا در نهایت در جمعیت نهایی فرد یا افراد برنده به عنوان پاسخ مسئله ارایه شود. اما به هر حال این الگوریتم تکاملی توانایی یافتن پاسخ نهایی را در بسیاری از موارد ندارد زیرا چند چالش وجود دارد. چالش اول وجود افزونگی بسیار زیادی است که در کل فضای جستجو وجود دارد و سربار محاسباتی را بشدت افزایش میدهد. چالش دوم ازریابی یک به یک برای هر زیر فضا است که کاری بسیار زمانبر است. چالش سوم در مورد الگوریتم های تکاملی شروع آن‌ها از پایه است و همین کار باعث می‌شود تا همگرایی آن‌ها کند اتفاق بیوفتد. برای حل این سه چالش از یک DAG پیشنهادی استفاده میکنیم تا افزونگی را تا حد ممکن از بین ببرد؟ سرعت همگرایی الگوریتم را افزایش دهد و میزان موازی سازی ارزیابی زیرفضا ها را تا حد ممکن افزایش دهد.

Reference DAG

ایده این مقاله در این مرحله این است که به جای پیدا کردن یک راه حل تکاملی از ابتدا از دانش پیشین یک مجموعه گراف برای افزایش سرعت استفاده کنیم. در این قسمت G^{ref} گراف DAG را نمایش میدهد که عمل‌کرد آن تأیید شده است. به عنوان معیار ارزیابی میزان نزدیکی DAG ها برای انتخاب بهترین آن‌ها از فاصله همینگ (Hamming Distance)استفاده میشود. بعد از هر عمل Mutation ما فاصله بین DAG تولید شده و DAG مرجع را از رابطه زیر محاسبه میکنیم:

$$r_H(G, G^{ref}) = \sum_{i,j} \frac{|G_{i,j} - {G^{ref}}_{i,j}|}{V(V-1)}.$$

همچنین یک حد آستانه T تعریف میکنیم و عمل Mutation را آنقدر انجام میدهیم تا فاصله hamming از حد آستانه کمتر شود.

Differentiable scoring function

به دلیل اینکه الگوریتم پیشنهادی به لایه‌های مختلف اجازه میدهد از ساختارهای مختلف سلولی استفاده کنند؟ روش‌های ارزیابی بر پایه ی سلول در این حالت غیر قابل استفاده هستند زیرا سایز فضای جستجو به صورت نمایی و انفجاری افزایش میابد و یک رابطه یک به یک بین کیفیت ساختار سلول و عمل‌کرد شبکه وجود ندارد. برای ارزیابی ساختار سلول‌های مختلف برای هر سلول مقدار تابع fitness را محاسبه میکنیم و با استفاده از بهینه سازی مبتنی بر گرادیان فرآیند جستجو را به صورت کامل مشتق پذیر کنیم. همانطور که در شکل کلی مدل پیشنهادی مشخص است در این روش جمعیت لایه‌ای G^l برای لایه ی l از شبکه اصلی را حفظ میکنیم. هر لایه با k تا ساختار سلولی نمونه برداری شده مقدار دهی اولیه میشود. خروجی هر ساختار سلولی نمونه برداری شده با توجه به امتیازی که از تابع fitness گرفته است با استفاده از رابطه زیر وزن میگیرد:

$$f_{S^l} = \sum_{k=1}^{K} p_k d_k(x) = \sum_{k=1}^{K} \frac{\alpha_k^l}{\sum_j \alpha_j^l} d_k(x),$$

امتیاز fitness برای K سلول انتخاب شده در هنگام آموزش Supernet با استفاده از backpropagation گرادیان بروز می‌شود تا خطای Cross-entropy را روی مجموعه آموزشی هدف مینیمم کند. برای کم کردن بروزرسانی گرادیان های نا همگون (imbalanced) روی امتیاز fitness در جمعیت هر سلول به واسطه ی نمونه برداری انتخاب تورنومنت (Tournament Selection) مقاله پیشنهادی با استفاده از مقیاس دهی امتیازها روی شماره تکرار آموزش مشکل بروزرسانی گام گرادیان را جبران میکند.

$$\frac{\partial L}{\partial \alpha_i^l} \approx \sum_{k=1}^{K} \frac{\partial L}{\partial g_k} p_k(\delta_{i,k} - p_i) \frac{n(d_i^l)}{n'(d_i^l)},$$

بعد از گذشت چند تکرار اولیه امتیاز fitness بروزرسانی شده در supernet برای بروزرسانی جمعیت در ساختار سلول‌ها استفاده میشود:

$$\alpha_k^{l\,(t)} = \epsilon \alpha_k^{l\,\star} + (1-\epsilon)\alpha_k^{l\,(t-1)},$$

نکته‌ی مهم این است که supernet در هر f تکرار از ابتدا دوباره تولید میشود.و همینطور در هنگام ساخت یک فضای جستجوی جدید امتیازهای Fitness و ساختارهای سلولی به صورت جفت جفت نمونه برداری می‌شوند که در ادامه هر امتیاز fitness متناظر با یک ساختار سلولی به صورت تکرار شونده بروزرسانی میشود.

جستجوی معماری ها در فضای تولید شده

حال پس از تولید فضای جستجو و تابع امتیاز دهی fitness که توضیح داده شد وقت آن است که در فضای تولید شده معماری مناسب حل مسئله را پیدا کنیم. فرآیند جستجو در فضای جستجو کافی است یک بار با مجموعه آموزشی هدف اجرا شود و فضای جستجوی تولید شده در این فرآیند میتواند توسط هر الگوریتم جستجویی مورد استفاده قرار گیرد. برای اثبات میزان تأثیر گذاری روش پیشنهادی (Auto Space) اجازه داده می‌شود که ساختار سلول‌های متفاوتی در لایه‌های مختلف مورد استفاده قرار گیرند و به صورت مستقیم روی مجموعه آموزشی ImageNet جستجو کنند تا مشکل در توانایی انتقال را حل کنند. در این قسمت برای افزایش سرعت آموزش و همینطور صرفه جویی در مصرف حافظه محاسباتی از روش پیشنهادی مقاله ENAS برای اشتراک وزن ها استفاده میکنیم که منجر به استفاده دوباره از وزن های بدست آمده در اجرای قبلی برای آموزش بهتر Supernet میشود. بعلاوه از ۳ معماری SOTA قبلی را بر اساس الگوریتم های جستجوی بر پایه گرادیان؟ بر پایه یادگیری تقویتی و نمونه برداری تصادفی استفاده میکنیم. تابع ضرر در این حالت برابر است با ترکیب تابع ضرر Cross-entropy و تابع خطی سازی MAdds:

$$Loss = Loss_{CE} + \lambda \text{MAdds}(N),$$

که MAdds تعداد عملگرهای MAdds در شبکه انتخابی میباشد.

راه حل بهبود مسئله (امتیازی):

در مقاله رفرنس [2] مراجع روشی نوین برای جستجوی مدل شبکه ارائه شده است. پیشنهاد این مقاله این است که یک پیش‌بینی کننده بر پایه شبکه عصبی به کار ببریم تا قدرت اکتشاف الگوریتم EA را برای NAS بهبود دهیم. این روش به NPENAS معروف است. برای پیش‌بینی کننده عصبی دو مدل پیشنهاد شده است. یک روش از روش های برپایه گراف برای تخمین همراه با عدم قطعیت استفاده میکند. و پیش‌بینی کننده دوم نیز همانند اولی یک معماری برپایه گراف است که میتواند به صورت مستقیم عملکرد مدل عصبی ورودی را تخمین بزند و در این حالت دیگر نیاز به ارزیابی یک به یک و یا غیر یک به یک که در توضیحات بالا اشاره کردیم نیست.

سوال ۲.

فرض کنید میخواهیم با استفاده از شبکه عمیق زیر بردار embedding مناسب برای یک مسئله را آموزش بدهیم. فرض کنید ۲ مجموعه سه تایی وارد این شبکه میشوند و خروجی آن مقادیر زیر است .

$$Triplet_1 = [A = (1,2), P = (4,2), N = (2,4)]$$
$$Triplet_2 = [A = (2,2), P = (3,2), N = (3,0)]$$

الف.

برای این داده ها مقدار تابع ضرر Triplet را با فرض α = 0.3 محاسبه کنید.

$$L(a,p,n) = max(0, D(a,p) - D(a,n) + \alpha)$$

$$D(p,q) = \sqrt{\sum_{i=1}^{n}(q_i - p_i)^2}$$

Triplet 1:

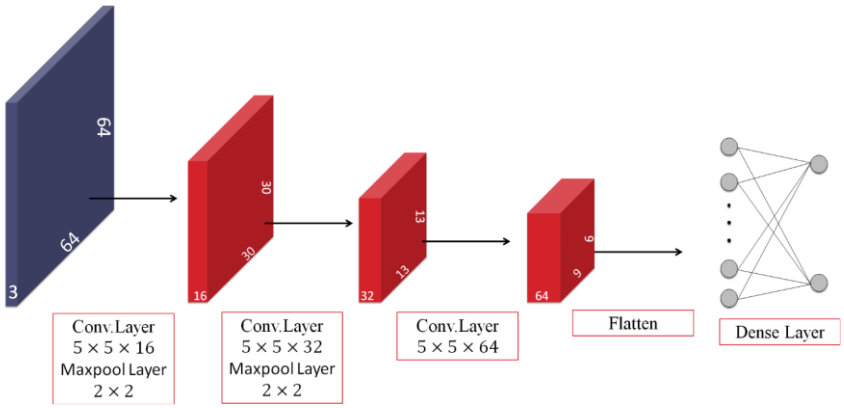
$$D(A,P) = \sqrt{(1-4)^2 + (2-2)^2} = \sqrt{9} = 3$$
$$D(A,N) = \sqrt{(1-2)^2 + (2-4)^2} = \sqrt{5}$$
$$L(a,p,n) = max(0, 3 - \sqrt{5} + 0.3) = 1.064$$

Triplet 2:

$$D(A,P) = \sqrt{(2-3)^2 + (2-2)^2} = \sqrt{1} = 1$$
$$D(A,N) = \sqrt{(2-3)^2 + (2-0)^2} = \sqrt{5}$$
$$L(a,p,n) = max(0, 3 - \sqrt{5} + 0.3) = 1.064$$

ب.

گرادیان این تابع ضرر نسبت به خروجی های شبکه را محاسبه کنید.



برای محاسبه Loss کلی، Triplet Loss ها را با هم جمع میکنیم.

$$Loss = \sum_{i=1}^{N} \left[\left\| f_i^a - f_i^p \right\|_2^2 - \left\| f_i^a - f_i^n \right\|_2^2 + \alpha \right] = 1.064 + 1.064 = 2.128$$

$$\frac{\partial Loss}{\partial f^a} = \sum_{i=1}^{N} \begin{cases} 2\,(f_i^n - f_i^p) & \text{if } \left(\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= 2(-2-2) + 2(0-2) = -8 - 4 = -12$$

$$\frac{\partial Loss}{\partial f^p} = \sum_{i=1}^{N} \begin{cases} -2\,(f_i^a - f_i^p) & \text{if } \left(\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= -2(-3+0) - 2(-1+0) = 6 + 2 = 8$$

$$\frac{\partial Loss}{\partial f^n} = \sum_{i=1}^{N} \begin{cases} 2\,(f_i^a - f_i^n) & \text{if } \left(\|f_i^a - f_i^p\|_2^2 - \|f_i^a - f_i^n\|_2^2 + \alpha \right) \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

$$= 2(-1-2) + 2(-1+2) = -6 + 2 = -4$$

سوال ۳.

در مسائلی که از Transfer Learning استفاده می‌شود، بسته به نوع مسئله می‌توان از دادگان برچسب خورده یا بدون برچسب در Source Domain استفاده کرد و سپس از دانش بدست آمده برای عملکرد بهتر در Target Domain بهره گرفت. اما زمانی که gap خیلی بزرگی بین Source Domain و Target Domain وجود داشته باشد، بسیاری از رو شهای رایج در حوزه Transfer Learning جواب قابل قبولی را به ما نمیدهند. به عنوان مثال، شکل سمت چپ برای مسئل های در حوزه دسته بندی اشیا در بین تصاویر است، اما شکل سمت راست مربوط به Task پیش بینی سطح فقر یک منطقه با استفاده از تصاویر ماهواره ای آن است در این شرایط معمولا استفاده از یک مدل از پیش آموزش دیده شده و حتی خارج کردن تعداد بیشتر ی لایه از حالت freeze دقت مورد نظر را به شما نخواهد داد. شما به عنوان یک متخصص در زمینه یادگیری عمیق چه راهکاری را برای حل این مسئله و استفاده حداکثری از دانش درون Source Domain پیشنهاد میکنید؟

در بسیاری از مسائل یادگیری عمیق نیاز است تا دانشی از یک یا چند Source Domain یاد بگیریم و سپس دانش را برای حل یک Target Domain استفاده کنیم. مشکلی که در صورت سوال به آن اشاره شده است عدم تطابق دامنه یا دامنه های Source با دامنه target است که به Domain Shift نیز معروف است. برای حل این مشکل روش هایی تحت عنوان Multi-source Domain Adaptation استفاده میشود. به طور کلی هدف این روش ها این است که با ترکیب و یا یادگیری فیچرهای ورودی مستقل از نوع Domain این مشکل را حل کنند. در مقاله رفرنس [3] مراجع مروری بر کارهای اخیر در این زمینه انجام گرفته است. به طور کلی برای مسائل Single Source Domain Adaptation از چهار راه حل کلی میتوان استفاده کرد:

۱. Discrepancy-based methods: تلاش میکنند ویژگی های دو دامنه را اندازه گیری اختلاف لایه های فعالسازی متناظر با هم تراز کنند.
۲. Adversarial generative methods: داده غیر واقعی تولید میکنند تا دو دامنه در سطح پیکسلی با هم تزار شوند.
۳. Adversarial discriminative methods: با استفاده از یک تابع معیار تفکیک کننده سعی در مینیمم کردن مقدار این تابع دارند که مینیمم شدن این تایع به معنی شباهت دو دامنه است.
۴. Reconstruction based methods: با ویژگی های استخراج شده از Source Domain سعی در بازسازی ورودی Target Domain دارند.

سوال ۴.

در این تمرین می‌خواهیم سیستمی را طراحی کنیم که توانایی دسته‌بندی داده‌های موجود در دیتاست CIFAR100 و پیدا کردن کمترین میزان خطا را به صورت خودکار داشته باشد. در صورت پیدا کردن بهترین مقدار برای هر Hyper Parameter مدل بهینه و مناسبی ساخته خواهد شد که توانایی دسته‌بندی داده‌ها با کمترین میزان خطا را دارد. در این تمرین برای استخراج بهترین مقدار هر Hyper Parameter میخواهیم از Optuna استفاده کنیم.

الف.

ابتدا با بررسی Optuna این فریم ورک را به طور کامل معرفی کنید.

Optuna یک فریم ورک بهینه ساز Hyperparameters است که به میتواند به راحتی روش های بهینه سازی state-of-the-art را پیاده سازی کند و عملیات بهینه سازی Hyperparameters را روی آن اجرا کند تا به سرعت به بهترین مدل حل مسئله دست پیدا کند. به صورت پیشفرض Optuna از الگوریتم بهینه سازی Bayesian (TPE) استفاده میکند البته الگوریتم های بهینه سازی دیگری نیز در این فریم ورک تعریف شده است که میتوان از آنها استفاده کرد.

Optuna الگوریتم های بهینه سازی اش را به دو دسته تقسیم میکند که عبارتند از:

- Sampling Strategy: الگوریتم هایی هستند که بهترین ترکیب پارامترها را با تمرکز بر مناطقی که Hyperparameters نتایج بهتری تولید کرده اند، انتخاب میکنند.
  - TPESampler(Tree-Structured Parzen Estimator):

یک الگوریتم بهینه سازی Bayesian است که :

۱. Hyperparameters را در ابتدا به صورت تصادفی انتخاب میکند و آنها را با توجه به score مرتب سازی میکند.
۲. Hyperparameters بر اساس چند کمیت از پیش تعریف شده به دو گروه تقسیم میشوند.
۳. با استفاده از kernel density estimators این دو گروه به تراکم های تخمینی l(x1) و g(x2) مدل میشوند.
۴. بهترین Hyperparameters با بالاترین بهبود انتخاب میشوند.
۵. Hyperparameters های انتخاب شده ارزیابی میشوند و دو باره مراحل ۲ تا ۵ را تکرار میشود و در اخرین تکرار بهترین Hyperparameters ها را گزارش میشود.

  - NSGAIISampler(Non-dominated sorting genetic algorithm):

یک الگوریتم بهینه سازی تکاملی است که چند تابع objective دارد.

در ابتدا لازم است مفهوم dominate بودن پاسخ A به پاسخ B را درک کنیم.

A بر Dominate B است اگر:

۱. A در هیچ Objective ای از B کمتر نباشد.
۲. A در حداقل یک Objective از B بهتر باشد.

الگوریتم به شکل زیر عمل میکند:

۱. یک جمعیت تصادفی ایجاد میشود و fitness تک تک آنها محاسبه میشود. در این جمعیت تمام افرادی که non-dominant (توسط هیچ فرد دیگری dominate نشده اند) را پیدا میکنیم و به آنها rank یک میدهیم و به باقی جواب ها rank دو میدهیم.
۲. دو نفر از جمعیت به صورت تصادفی انتخاب میشوند و جواب بهتر parent 1 میشود.
۳. دو parent را با هم ترکیب میکنیم (Recombination). فرزند آنها با استفاده از mutation چند تغییر در مقادیر خود میدهد. و این کار را تا دو برابر شدن جمعیت اولیه تکرار میکنیم.
۴. جمعیت جدید بر اساس nondomination مرتب سازی میشوند.
۵. مراحل ۲ تا ۵ تکرار میشوند تا در انتها با استفاده از این الگوریتم تکاملی بهترین فرد انتخاب شود. لازم به ذکر است منظور از هر فرد یک جواب است. (نیاز است تا بتوانیم هر جواب احتمالی مسئله را به صورت یک فرد در جمعیت مدل کنیم.)

- Pruning Strategy: این روش ها بر پایه ی Early-stopping هستند که سعی میکند هنگام رسیدن به یک نقطه مناسب، فرآیند جستجوی Hyperparameters را لغو کند.
  - SuccessiveHalvingPruner(Asynchronous Successive Halving):
۱. در ابتدا به صورت تصادفی مقادیر Hyperparameters را مقدار دهی اولیه میکنیم.
۲. به تعداد ۱ epoch، Trials را آموزش میدهیم تا به تعداد حداکثری trials برسیم.
۳. در یک زمان، یک tiral یک مرتبه ارتقا پیدا میکند (همانند Rank در الگوریتم بالا) و با تعداد epoch های بیشتری آموزش میبینند.

ب)

مدل پیشنهادی باید شامل تعدادی لایه Convolution و Fully Connected باشد و در صورت نیاز می‌توان از لایه Pooling و Dropout استفاده نمود. در جدول زیر بخشی از اطلاعات مورد نیاز برای tune مدل ارائه شده است. با استفاده از optuna و تعریف بازه مناسب، مقدار بهینه را برای هر Hyper parameter بدست آورید. همچنین با توجه به دانش خود مدل را تا حد امکان بهبود دهید.

کد این قسمت در کنار گزارش با نام CIFAR100_Optuna_PyTorch_Q4_B_Epoch_53 موجود میباشد. دلیل این که ۵۳ epoch بیشتر اجرا نشده است تمام شدن زمان مجاز استفاده از Colab است.

ج)

ابتدا pruning را توضیح دهید و سپس با تغییر استراتژی بهینه سازی و استفاده از pruning سعی کنید مقادیر بهینه را بدست بیاورید. آیا دقت مدل بهینه نسبت به قسمت قبل کاهش پیدا کرد. تحلیل خود را درباره استفاده از pruning بیان کنید.

در مورد pruining در قسمت الف همین سوال توضیحاتی داده شده است. در مورد نتایج حاصل شده میتوان اظهار داشت با توجه به مجموعه آموزشی صورت سوال (CIFAR100) که یک مجموعه آموزشی به نسبت بزرگ است و همینطور با توجه به مقادیر حداکثری تعداد لایه های کانولوشنی و کاملا متصل و تعداد نرون ها در هر لایه در هیچ حالتی نمیتوان مدلی تولید کرد

که ظرفیت یادگیری این مجموعه آموزشی را داشته باشد و تمام مدل های قسمت **ب**، **ج** و **د** از دقت های پایین و نزدیکی برخوردارند. کد مربوط به این قسمت با نام CIFAR100_Optuna_pytorch_Q4_C_Epoch_415 موجود میباشد. اجرای کد میتوانست بیشتر از epoch 415 به طول بیانجامد اما با محدودیت زمانی مواجه شدم.

د)

در این بخش علاوه بر استفاده از استراتژی pruning ، تعداد trial ها را ۱۰۰ در نظر بگیرید. آیا دقت مدل بهینه نسبت به قسمت قبل کاهش پیدا کرد. تحلیل خود را درباره محدود کردن تعداد دفعات تکرار آزمایش بنویسید.

کد مربوط به این قسمت نیز با نام CIFAR100_Optuna_pytorch_Q4_D_Epoch_43 ضمیمه شد. نتیجه ای که از نتایج میتوان گرفت به دلیل پایین بودن ظرفیت مدل بالا بردن تعداد epoch ها از ۱۰ به ۱۰۰ فقط زمان آموزش را بشدت افزایش میدهد و هیچ تفاوتی در عملکرد مدل ندارد.

ه)

برای شما به عنوان یک متخصص در زمینه یادگیری عمیق، در این مسئله کدام Hyper parameter از اهمیت بالاتری برخوردار است و اگر بخواهید هر Hyper parameter را به صورت جداگانه بهینه کنید، اولویت بهینه سازی را به آن میدهید. ترتیب اولویت خود را بنویسید و دلایل انتخاب این اولویت را بیان کنید.

اولویت اول : تعداد لایه های کانولوشنی و کاملا متصل:

دلیل این انتخاب این است که تعداد لایه ها اثر مستقیم بر ظرفیت یادگیری مدل دارد. استفاده از بهترین hyperparameter های دیگر اگر مدل به طور کلی ظرفیت نداشته باشد و یا ظرفیت مدل بیشتر از حد مورد نیاز باشد به راحتی دچار underfitting و overfitting خواهیم شد.

اولویت دوم: بهینه ساز و نرخ آموزش

هرچند که نوع بهینه ساز و learning rate دو Hyperparameter جدا هستند اما هیچکدام جدا از دیگری به تنهایی نمیتواند عملکرد مدل را به طور چشمگیری بهبود دهد. حتی در بهینه سازی مانند Adam که سعی میکند نرخ آموزش را به صورت خودکار تنظیم کند باز هم مقدار اولیه نرخ آموزش تاثیر بسیار زیادی خواهد داشت. و در طرف مقابل اگر بتوان به نحوی بهترین نرخ آموزش را در هر مرحله از آموزش مدل پیدا کرد اما بهینه ساز نتواند در یافتن نقطه بهینه خوب عمل کند نتایج مدل موثر نخواهند بود.

اولویت سوم: جزئیات مربوط به لایه ها (تعداد نرون، اندازه کرنل، تعداد فیلترها)

اگر دو اولویت اول و دوم را به خوبی تنظیم کنیم مدل با انتخاب یک hyperparameter نسبتا خوب (نه صرفا بهینه) در بین hyperparameter های اولویت سوم میتواند نتایج بسیار خوبی روی مجموعه آموزشی بگیرد.

مراجع

1) towardsdatascience.com/machine-learning-hyperparameter-optimization-with-optuna

2) Wei, C., Niu, C., Tang, Y., Wang, Y., Hu, H., & Liang, J. (2022). Npenas: Neural predictor guided evolution for neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*.

3) Zhao, Sicheng, et al. "Multi-source domain adaptation in the deep learning era: A systematic survey." *arXiv preprint arXiv:2002.12169* (2020).